A Framework for Intrusion Detection based on Frequent Subgraph Mining

Vitali Herrera-Semenets* Niusvel Acosta-Mendoza * Andrés Gago-Alonso *

Abstract

In many application contexts, graphs have been widely used to model data due to their expressiveness and their suitability, since some kind of entities and their relationships must be encoded in the same structure. Nowadays, the progress of the technologies of computer and communications has boosted the appearing of unusual or suspicious activity, affecting the network operations. In this paper, a framework for intrusion detection based on frequent subgraph mining is presented. The former scheme includes a novel strategy for reducing the volume of data to be processed, without losing useful information. Finally, our proposal, using different settings, is analyzed and compared regarding some state-of-the-art approaches, achieving good results.

Keywords: intrusion detection, feature selection, graph mining, classification.

1 Introduction

New technologies have enabled the use of the network, increasing the flow of information. This progress also brings in malicious users known as intruders. Intruders achieve their goals by exploiting weaknesses or backdoor's in vulnerable systems, unpatched ones, or systems infected by trojans. The need to alert this kind of attack boost up the use of intrusion detection systems (IDS), allowing it to prevent or reduce the damage caused by intruders.

In this paper, a framework for intrusion detection is presented. The proposal consists of two stages: training and classification. Meanwhile, the training stage includes four modules: preprocessing, graph mining, representation, and training. Preprocessing module is focused on reducing the volume of input data, without losing useful information by combining two already reported feature selection algorithms [7, 6]; here, the values of each selected features are clustered and relabeled. In the graph mining module, each transaction is represented as a graph. Thus, the training collection becomes a graph collection, which is mined for finding a set of frequent subgraphs. Next, in the representation module, each transaction is represented by a feature vector, taking into account the frequent subgraphs. Finally, the resulting set of feature vectors is used, in the training module, for obtaining the classification model.

The features and labels calculated during training are used in the classification stage, for obtaining the graph of each upcoming unclassified transaction. Next, the resulting graph is contrasted regarding the already calculated frequent subgraphs, obtaining the corresponding feature vector. Finally, this vector is classified according to the classification model.

Some state-of-the-art approaches are analyzed and compared regarding our proposal, using different settings. This paper is organized as follows. The related works are presented in Section 2. Our approach is introduced in Section 3. The achieved experimental results using a supervised intrusion detection dataset are shown in Section 4. In Section 5, the significance and impact of the proposed framework, as well as its possible application in other domains, are analyzed. Finally, the conclusions and an idea for future work are outlined in Section 6.

2 Related works

In the last years, some intrusion detection methods are proposed [20, 21, 22, 25]. In this section, an overview of these kind of methods is presented, including only those which were applied on the KDD dataset [4].

A rule-based framework PNrule is proposed by Agarwal and Joshi [14], where the main idea is to learn a rule-based model using two sets of rules. The first set of rules is used to predict the presence of a class. The other set of rules is used to predict the absence of a class. Depending on which combination of rules applies, they predict the record to be in a particular class with certain score in the interval $[0\%, 100\%]$. This score represents the probability of the record belongs to the target class.

Neural networks can be applied in intrusion detection context [24]. The technique presented by Ahmad [19] uses a Principal Component Analysis (PCA) and genetic algorithm for the feature selection. The feature set obtained by this process is presented to a Multilayer Perceptron (MLP) for classification purpose. In the training phase the weights of the system are updated by carrying out a certain steps. The testing phase involves two steps: verification step and generalization

*Advanced Technologies Application Center. 7a # 21406, Playa, C.P. 12200, Havana, Cuba - CENATAV - {vherrera,nacosta,agago}@cenatav.co.cu

step. In the verification step, a trained system is tested with the data used in training. This step evaluates how well the trained system has learned the training patterns in the training dataset. In generalization step, testing is conducted with data that were not used in training, to evaluate generalization ability of the trained network.

A multi-classifier model is introduced by Sabhnani and Serpen [15]. This model is created using most promising classifiers for a given attack category. The best algorithms identified for each category were: MLP, k-means and Gaussian.

An approach based on a hierarchy of self-organizing feature maps (SOMs) has been proposed by Kayacik *et al.* [16]. The training process consists of two stages. The first step is for the general organization of the SOM and the second for the fine-tuning of neurons. The training process is repeated for each SOM comprising the hierarchy. In this method, the best performance is achieved using a two-layer SOM hierarchy, based on all 41 features from the KDD dataset.

Revathi and Malathi [26] presents a hybrid system to reduce the dimensionality of the data and classify network intrusions. The hybrid system is based on a technique called Simplified Swarm Optimization (SSO) for the feature selection and a data mining algorithm Random Forest as the classifier.

Another method is proposed by Jeya *et al.* [10], which consists in a classification method using a discriminant analysis. First, the authors make a feature selection. Next, the data collection is split into four subsets, each one containing a specific attack category and normal connections. Then, a discriminant analysis is applied using SPSS tool [11] on a reduced feature set. Finally, a classification summary for each category is presented.

There is an approach based on anomaly detection using graphs. The method proposed by Noble and Cook [9] is focused on detecting specific and unusual substructures within a graph. The idea is representing the data collection as a graph and splitting it into distinct subgraphs, determining the anomalous subgraphs regarding the remaining ones.

There are few studies based on graphs for intrusion detection [9, 27, 28]. They are commonly focused on detecting anomalies and associate them to intruders. This fact can lead a poor system performance; since, as it is discussed in Section 4.2, an intruder is not considered an anomaly. Moreover, the reduction of data collection is usually performed only by the selection of features. Ignoring other details such as data redundancy, which can affect the efficiency of the system. Our framework takes into account these details and processes the reduced collection using a frequent subgraph mining [1] algorithm.

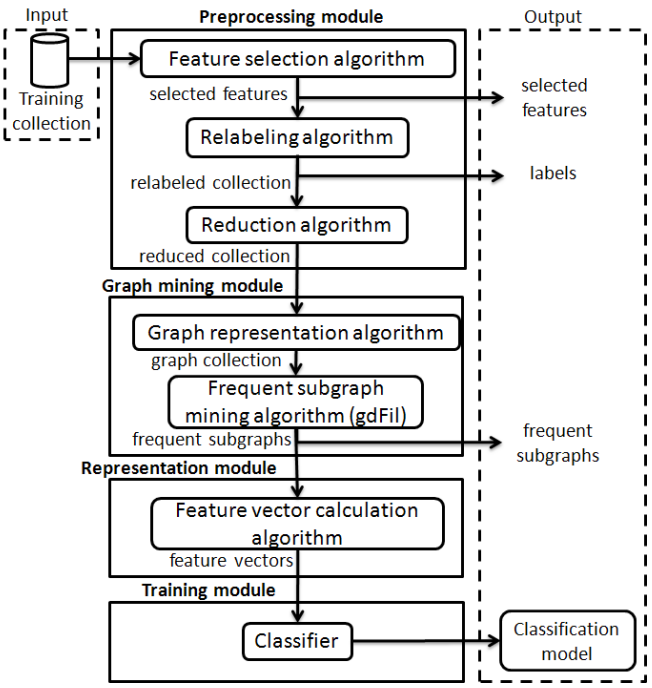

Figure 1: Training stage diagram.

3 Proposed method

In this section, the proposed framework for intrusion detection based on frequent subgraph mining is presented. Our approach comprises of two stages: training and classification.

3.1 Training stage. This stage is composed by four modules: preprocessing, graph mining, representation, and training (see Figure 1). The preprocessing module processes the training collection, starting with a feature selection algorithm. The feature selection is performed to reduce the dimensionality of the input data. Applying feature selection we obtain a subset containing the most representative features according to the outcome scores. In this way, only the representative features are used as attributes for classification.

Two selection techniques are used for identifying the representative feature set. One of these techniques is Univariate Feature Selection [6], where a statical test based on analysis of variance (ANOVA) is applied to each feature individually. The other technique is support vector machine feature weights (SVM Weight) with a linear kernel [7], where the normalized feature weights are used to sort the features according to its relevance.

Once obtained the scores of both selection methods,

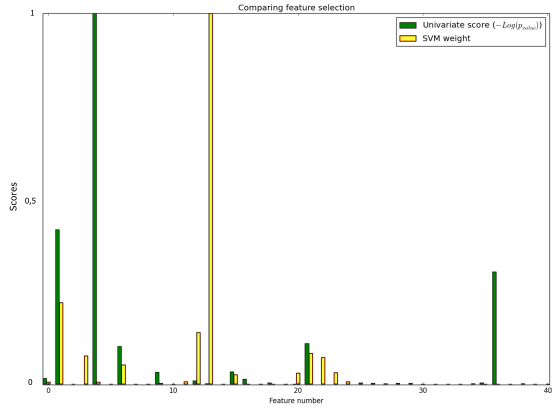

Figure 2: Comparing feature selection algorithms.

they are normalized between [0, 1]. Let s_{max} be the maximum score of S_{SVM}, for the SVM Weight algorithm the obtained scores set S_{SVM} is normalized following the equation (3.1):

$$N_{SVM} = \left\{ n_i \mid s_i \in S_{SVM}, n_i = \frac{s_i}{s_{max}} \right\} \tag{3.1}$$

where N_{SVM} is the normalized set of values for S_{SVM}.

In the same way, the obtained scores set S_{UV} from the Univariate algorithm is normalized according to the equation (3.2):

$$N_{UV} = \left\{ n'_i \mid s'_i \in S_{UV}, n'_i = \frac{s'_i}{s'_{max}} \right\} \tag{3.2}$$

where N_{UV} is the normalized set of values for S_{UV} and s'_{max} is the maximum score obtained in S_{UV}. In Figure 2, normalized scores by the SVM Weight algorithm and the univariate algorithm are represented for each feature. Note that the features are shown from 0 to 40, being 41 the number of features in KDD dataset.

For each obtained normalized set (N_{SVM} and N_{UV}) is computed their scores mean ($\overline{N}_{SVM}$ and $\overline{N}_{UV}$). Then, a representative feature is that where $n_i > \overline{N}_{SVM}$ or $n'_i > \overline{N}_{UV}$. In this way, two sets of features are obtained: F_{SVM} set of features from SVM Weight algorithm and F_{UV} set of features from Univariate algorithm. Finally, the set of features $F_f = F_{SVM} \cup F_{UV}$ is identified as the most representative set.

Once the features have been selected by the selection algorithm, new labels for these features are generated by a relabeling algorithm. This relabeling algorithm assigns a unique numerical label for each non numerical feature value. Furthermore, using k-means algorithm [8], the relabeling algorithm creates clusters for numerical features. Each obtained cluster comprises a range of numerical values, which are represented by a unique numerical label. The use of these clusters allows us to cover values that do not exist in the training dataset and are included in the classification stage.

For example, suppose that the feature f_1 takes the values $0.0, 0.2, 0.3, 0.6, 0.7$ and 1.0 in the training stage, and a cluster $c = [0.7, 1.0]$ is obtained. Then, in the classification stage, a new transaction in which the feature f_1 takes the value 0.9, where this value not appear in the training stage; however, it is possible to represent it using the label assigned to the cluster c.

In order to reduce the values dimensionality of the numerical features, we seek a small value for k (20 in our case) that allows us to keep the representativeness of data. Finally, it is important to note that a restriction for our clustering method is added (see equation 3.3)

$$k = \begin{cases} |f_i| & if\ |f_i| < k \\ n & otherwise \end{cases} \tag{3.3}$$

where f_i is a numerical feature and n is the default value of k (20 in our case).

The next step is to apply the reduction algorithm. This algorithm involves removing duplicate transactions from the relabeled training collection. The reduction process is performed due to duplicate transactions do not provide useful information. Notice that a transaction is a duplicate transaction if there exist at most other transaction with the same feature values and class. This algorithm solves the problem reported by Tavallaee *et al.* [29], related to redundant records in the KDD Cup '99 dataset.

In the graph mining module, the collection obtained by the reduction algorithm is processed by a graph representation algorithm. The purpose of this representation algorithm is to represents each transaction as a star graph [9]. In a graphical representation, a star graph has a core vertex representing the transaction itself and a vertex for each corresponding feature connected to the core (see Figure 3). The edge label is the feature index, and the vertex label is the value of the corresponding feature.

Here, a method based on identifying frequent item sets can be used. In case of using this method, a collision between labels that belong to different features may appear. This is due to there are no differences between labels from different features, because they represent the same value. Thus, we would have to generate different labels for each feature. Moreover, when the number of existing features $|F|$ is significantly higher, then a considerable increment of the number of the generated labels η could be obtained; since $\eta = |F| \cdot k$, where k represents the number of labels to be obtained for each

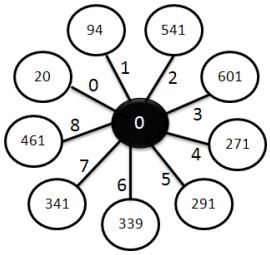

Figure 3: Star graph example.

feature.

The graph-based representation allows us to eliminate such collisions. For example, suppose that we have two features f_1 and f_2, and both have a label 2, then using graph-based representation each label has a different significance; since the edges indicate the corresponding feature. Next, the obtained graph collection is processed by the gdFil algorithm [2] for frequent subgraph mining.

Any other graph miner could be used instead of gdFil. However, gdFil get good execution time regarding gSpan [3] and other graph miners, since it performs a full complete duplicate candidate pruning [2].

The frequent subgraphs [1] obtained by gdFil algorithm are represented as feature vectors by the feature vector calculation algorithm. A vector $v_G = [g_1, g_2, \ldots, g_j]$ contains elements, where j is given by the number of frequent subgraphs detected. The value of each element g_l (where $l = 1, 2, ..., j$) is binary. Thus, one element $g_l = 1$ indicates that the subgraph represented by g_l is contained in the graph G represented by v_G.

Finally, in the training module, the feature vectors are processed by a classifier, and a classification model is obtained.

3.2 Classification stage. This stage (see Figure 4) contains three modules: preprocessing, pattern identification and representation. The selected features and labels obtained during the training stage are used in the preprocessing module to represent a new unclassified transaction.

In the patterns identification module, the relabeled transaction is processed by the same graph representation algorithm that was used in the training stage; only this time, the result is a single graph. Next, a pattern search algorithm is performed. This algorithm contrast the resulting graph regarding the already calculated frequent subgraphs.

In the representation module, the feature vector calculation algorithm is used. Thus, the identified subgraphs by the pattern search algorithm are represented as the corresponding feature vector of the processed

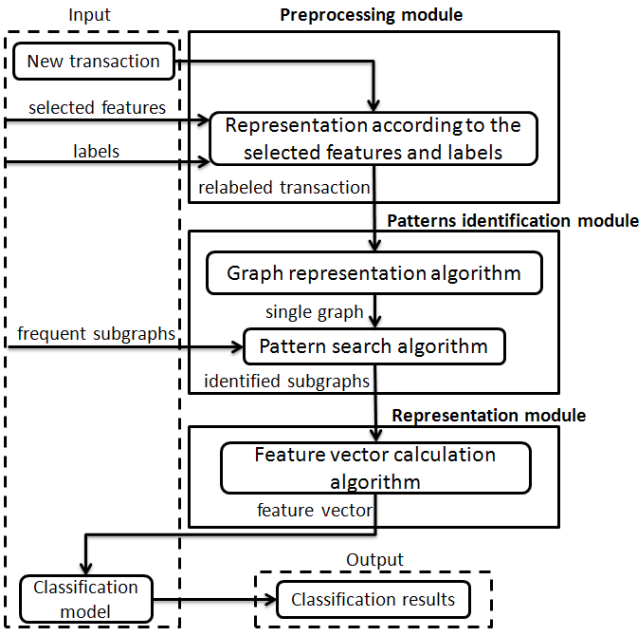

Figure 4: Classification stage diagram.

graph. Finally, this vector is classified according to the classification model.

4 Experiment

In this section, the dataset used for evaluating the proposed framework is analyzed and the classification results achieved are presented.

4.1 KDD Cup '99 dataset [4]. This supervised intrusion detection dataset is used in our experiment. The dataset provides connection records generated by a simulation of a military network. Such dataset contains two labeled collections: training with 4 898 431 connection records (transactions) and testing with 311 029 transactions. In the training collection, the number of classes is 23 (22 different attack types and "normal" label). In the testing collection, the number of classes is 38 (37 different attack types and "normal" label). In the KDD'99 contest, the attack classes are clustered in the following four categories.

1. Probe: surveillance and other probing.
2. DoS: denial of service.
3. U2R: unauthorized access to local superuser (root) privileges.
4. R2L: unauthorized access from a remote machine.

Therefore, it can be seen as a database with samples of four attack categories, unlike other databases like

the CAIDA "DDoS Attack 2007" dataset [5], which only has one attack category (DoS). Each transaction in the KDD'99 dataset contains 41 features describing it, where these features can be continuous or discrete.

4.2 KDD Cup '99 dataset processed by others methods. In this section, the ways of processing the KDD dataset by some methods are discussed.

In [10], they split the training collection into 4 attack subsets. Each subset is analyzed using a correlation analysis for identifying the important features for a specific attack. This analysis result gives a set of features for each subset. These features are considered as relevant features for each attack. When the data collection is partitioned into subsets for each attack category and each subset is analyzed independently. This approach becomes in a two class categorization problem, instead of the five classes (Probe, DoS, U2R, R2L).

In [9], it is assumed that an anomaly is an event that occurs rarely. Therefore, the problem appears when a collection of data, where attacks are quite common, is processed. Since when attacks are common, they cannot be classified as anomaly, affecting the performance of anomaly detection systems. To solve this problem, they build new collections using the test collection. In each new collection, most of the selected transactions (96%-98%) belong to the "normal" class, while the rest belong to one attack type.

In the methods [10] and [9], the training and test collection are split into subsets to adapt them to their proposal. Each subset is processed independently. Particularly in [10], they use different sets of features for each subset. The method proposed in this paper processes the complete test collection without partitioning it, and uses a single set of selected features.

4.3 Experimental results. Experiments were performed on a machine with 4 GB of RAM and a Quad Core processor at 2.5 GHz. The algorithms are implemented using ANSI-C programming language.

The selected features during the training process were $F_{UV} = \{1, 4, 36\}$ and $F_{SVM} = \{1, 3, 6, 12, 13, 21, 22\}$. The final set was $F_f = \{1, 3, 4, 6, 12, 13, 21, 22, 36\}$. Table 1 shows the selected features, their position in the original transaction (starting with 0), a short description about them and their type.

A comparison of the features selected by our approach regarding the features used in other works is presented in Table 2. For example, our features selected match with the most relevant features selected by Kayacik *et al.* [13] for some individual classes as is shown in Table 2.

Table 2: List of features used by our approach and the approach reported in [13] for specific classes.

Class	**Relevant Features**
normal	36
smurf	1, 4, 22
neptune	3
land	6
back	12
buffer_overflow	13
warezclient	21

Table 3: List of features used by our approach and the approach reported in [12].

Attack category	**Relevant Features**
DoS	3, 4, 12, 22, 36
Probe	3, 4
U2R	1, 4, 12, 13
R2L	4, 21, 22

Another comparison between our selected features according to the categories and the ones used in [12], is shown in Table 3.

After relabeling, the training collection represented with the selected features is processed by the reduction algorithm. This algorithm allows us to reduce the amount of transactions from 4 898 431 to 815, being a reduction over 98%. A meaningful dimensionality reduction, achieved by our proposal in terms of the number of transactions for each category, can be seen in Table 4.

When the gdFil algorithm with $\delta = 0.4$ was performed, the feature vectors obtained in the representation module, are processed by three different classifiers. The parameter δ represents the support threshold [2]. The three used classifiers are executed on the Weka platform [23], and they are: J48graft, classification via Regression and SMO. In Table 5, the accuracy achieved by different approaches over the same KDD'99 test collection is shown. The classifiers used in our approach are marked with " * ".

The outcomes of each classifier used in the framework have a category where the accuracy is higher than the other compared methods. Using a J48graft classifier the normal class is very well classified; however, in the attack categories the expected results are not obtained. When the SMO classifier and the Regression classifier are used, the maximum accuracy for R2L category is obtained, tripling the results reported in most of the remaining approaches. Instead, in the rest of the categories the expected results are not obtained.

Table 1: List of the features selected by our proposal.

Id	Feature	Description	Type
1	protocol type	Connection protocol (e.g. tcp, udp)	Discrete
3	flag	Status flag of the connection	Discrete
4	source bytes	Bytes sent from source to destination	Continuous
6	land	1if connection is from/to the same host/port; 0 otherwise	Continuous
12	compromised	number of compromised conditions	Continuous
13	root shell	1 if root shell is obtained; 0 otherwise	Continuous
21	is guest login	1 if the login is a guest login; 0 otherwise	Continuous
22	count	number of connection to the same host as the current connection in the past two seconds	Continuous
36	dst host srv diff host rate	% of connections to the same service coming from different host	Continuous

Table 4: Comparison between the original training collection and reduced training collection.

	Original training collection		**Reduced training collection**	
Categories	Num. of trans.	Percent of total	Num. of trans.	Percent of total
normal	972781	19,9 %	478	58,7 %
DoS	2883370	79,3 %	116	14,2 %
Probe	41102	0,84 %	156	19,1 %
U2R	52	0,001 %	17	2,08 %
R2L	1126	0,023 %	48	5,89 %

The experimental results show that the proposed framework, which train with a data collection reduced by 98%, is able to achieve good results in specific categories depending on the used classifier.

5 Significance and Impact

Many datasets can be represented by collections of graphs, which can be classified by the real data structures. The discovery of frequent patterns, especially the detection of frequent subgraphs in graph collections, has boosted its use in many application on different domains of science. As it was shown in this paper, the transactions generated by network traffic can be represented as a graph collection. This representation allows applying the frequent subgraph mining in intrusion detection tasks and opening a new path for future researches.

For detecting intruders, processing large volumes of data generated by network traffic is required. In these cases, the computational performance in terms of time and storage capacity is affected. Our approach provides a solution to this problem, including a preprocessing module, which drastically reduces the dimensionality of data.

In this paper, the proposed framework is designed for the intrusion detection. However, it can be extended for the fraud detection domain in areas where the data can be represented as graphs, such as: in telecommunications, banks, online auctions, among others. Observing the results obtained by our framework using J48graft, in the categorization problem on KDD Cup '99 with five classes; the 99.9% of transactions in the normal class were correctly classified. In the fraud detection, which usually consists of a two classes (fraud, not fraud) categorization problems, could be obtained encouraging results.

6 Conclusion

There are a wide variety of techniques proposed for intrusion detection based on different approaches, which were analyzed in this paper. Some of these techniques, in the best scenario, use the feature selection for reducing the dimensionality of the data. Our proposal includes an initial step, which is very important to deal with large volumes of data. The preprocessing module drastically reduces the training collection, without losing useful information. To achieve this reduction, three specific algorithms are used: the feature selection algorithm, relabeling algorithm and reduction algorithm.

Using the training collection as a matrix, where rows are transactions and columns represent features, a matrix of 4 898 431 rows and 41 columns is initially given. After being processed by our framework, the

Table 5: Accuracy achieved by different approaches on the entire "corrected" test collection.

Approach	**normal**	**Probe**	**DoS**	**U2R**	**R2L**
*J48graft	99,9 %	43,73 %	71,41 %	0 %	0 %
*Regression	55,09 %	51,44 %	71,44 %	4,39 %	27,7 %
*SMO	54,97 %	53,29 %	71,6 %	3,95 %	28,2 %
KDD '99 winner [17]	99,5 %	83,3 %	97,1 %	13,2 %	8,4 %
KDD '99 runner-up [18]	99,4 %	84,5 %	97,5 %	11,5 %	7,3 %
PNrule [14]	99,5 %	73,2 %	96,9 %	6,6 %	10,7 %
SOM [16]	98,6 %	81,3 %	96,9 %	0 %	1,1 %
Multi-Classif. [15]	-	88,7 %	97,3 %	29,8 %	9,6 %

result was a matrix of 816 rows and 9 columns. This result makes it possible to reduce the execution time required for training, as well as the noise generated by features and transactions that do not provide useful information.

After analyzing the behavior of our method on a five class categorization problem, we plan to apply it to a binary categorization problem. The domain of fraud detection turns out to be a good candidate to test our method, where transactions would be classified in fraudulent or legitimate.

References

[1] J. Huan, W. Wang, and J. Prins. *Efficient mining of frequent subgraphs in the presence of isomorphism*, In proceedings of the Third IEEE International Conference on Data Mining, pp. 549–552, Washington, DC, USA, 2003.

[2] A. Gago-Alonso, J. A. Carrasco-Ochoa, J. E. Medina-Pagola, and J. F. Martnez-Trinidad. *Full Duplicate Candidate Pruning for Frequent Connected Subgraph Mining*, Integrated Computer-Aided Engineering, 17(3):pp. 211–225, 2010.

[3] X. Yan and J. Han. *gSpan: Graph-Based Substructure Pattern Mining*, In Proceedings of the 2002 International Conference on Data Mining (ICDM'02), pp. 721–724, Maebashi, Japan, 2002.

[4] KDD Cup 1999 Computer network intrusion detection. [Online]. Available: http://sigkdd.org/kdd-cup-1999-computer-network-intrusion-detection.

[5] The CAIDA UCSD "DDoS Attack 2007" Dataset. [Online]. Available: http://www.caida.org/data/passive/ddos-20070804_dataset.xml

[6] Univariate feature selection. [Online]. Available: http://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection.

[7] Yin-Wen Chang and Chih-Jen Lin. *Feature Ranking Using Linear SVM*. JMLR: Workshop and Conference Proceed, 3:pp. 53–64, 2008.

[8] J. A. Hartigan and M. A. Wong. *Algorithm AS 136: A K-means clustering algorithm*. Applied Statistics, Royal Statistical Society, pp. 100–108, 1979.

[9] C. C. Noble and D. J. Cook. *Graph-based anomaly detection*. In proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 631–636, Washington, DC, USA, 2003.ACM.

[10] P. G. Jeya, M. Ravichandran and C. S. Ravichandran. *Efficient Classifier for R2L and U2R Attacks*. International Journal of Computer Applications , 45(21):pp. 28–32, Washington, DC, USA, 2012.

[11] SPSS 13.0 Base User's Guide. [Online]. Available: http://sscnet.ucla.edu/labs/SPSS13/SPSSBaseUsersGuide13.0.pdf.

[12] S. O. Al-mamory and F. S. Jassim. *Evaluation of Different Data Mining Algorithms with KDD CUP 99 Data Set*. Journal of Babylon University, 21(8):pp. 2663–2681, 2013.

[13] H. G. Kayacik, A. N. Zincir-Heywood and M. I. Heywood. *Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets*. In proceedings of the third annual conference on privacy, security and trust, 2005.

[14] R. Agarwal and M. V. Joshi. *PNrule: A New Framework for Learning Classifier Models in Data Mining*. Technical report, Department of Computer Science, University of Minnesota, USA, 2000.

[15] M. Sabhnani and G. Serpen. *Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context*, In MLMTA, pp. 209–215, 2003.

[16] H. G. Kayacik, A. N. Zincir-Heywood and M. I. Heywood. *A hierarchical SOM-based intrusion detection system*. Engineering Applications of Artificial Intelligence, 20(4):pp. 439–451, 2007.

[17] B. Pfahringer. *Winning the FDD99 classification cup: bagged- boosting*. SIGKDD Explorations, ACM SIGKDD, 1(2):pp. 65-66, 2000.

[18] I. Levin. *KDD-99 Classifier Learning Contest*. LLSoft Reslts Overview, SIGKDD Explorations, ACM SIGKDD, 1(2):pp. 67-75, 2000.

[19] I. Ahmad. *Enhancing MLP Performance in Intrusion Detection using Optimal Feature Subset Selection based*

on Genetic Principal Components. International Journal of Applied Mathematics & Information Sciences, 8(2):pp. 639–649, 2014.

[20] A. Srivastav, P. Kumar, R. Goel. *Evaluation of Network Intrusion Detection System using PCA and NBA*. International Journal of Advanced Research in Computer Engineering & Technology, 2(11):pp. 2873–2881, 2013.

[21] L. Xiao, Y. Chen, and C. K. Chang. *Bayesian Model Averaging of Bayesian Network Classifiers for Intrusion Detection*. In Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International, pp. 128–133, 2014.IEEE.

[22] P. Amudha, S. Karthik and S. Sivakumari. *Classification Techniques for Intrusion Detection An Overview*. International Journal of Computer Applications, 76(16):pp. 33–40, 2013.

[23] Weka 3: Data Mining Software in Java. [Online]. Available: http://cs.waikato.ac.nz/ml/weka/

[24] G. Liu, Z. Yi and S. Yang. *A hierarchical intrusion detection model based on the PCA neural networks*. Neurocomputing, Advances in Computational Intelligence and Learning, 14th European Symposium on Artificial Neural Networks, 70:pp. 1561–1568, 2006.

[25] R. Singh and D. Singh. *A Review of Network Intrusion Detection System Based on KDD Dataset*. International Journal of Engineering and Technoscience, 5(1):pp. 10–15, 2014.

[26] S. Revathi and A. Malathi. *Network Intrusion Detection Using Hybrid Simplified Swarm Optimization and Random Forest Algorithm on Nsl-Kdd Dataset*. International Journal of Engineering and Computer Science, 3(2):pp. 3873–3876, 2014.

[27] J. Tölle and O. Niggemann. *Supporting intrusion detection by graph clustering and graph drawing*. In Proceedings of 3rd International Workshop on Recent Advances in Intrusion Detection RAID, 2000.

[28] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip and D. Zerkle. *GrIDS-a graph based intrusion detection system for large networks*. In Proceedings of the 19th National Information Systems Security Conference, 1:pp. 361–370, 1996.

[29] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani. *A detailed analysis of the KDD CUP 99 data set*. In Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications, 2009.